caWorkBench 2.0 Use Cases

Andrea Califano
Kenneth Smith
Manjunath Kustagi
Stuart Fischer

Use Case 1A – Searching for upstream conserved motifs across co-regulated genes (before caWorkbench 2.0).

The researcher loads microarray gene expression data from local files into his favorite analysis package and runs filtering, normalization, and perhaps differential expression routines. He may then need to transfer the data via a file to a separate program for advanced analysis such as clustering or SOM (self-organizing maps) to find groups of genes with similar expression patterns. An interesting group is chosen and a list of genes is exported to a file. To retrieve the upstream sequences for the genes on this list, the user may first need to translate the gene names exported by the analysis program to a synonymous name such as the RefSeq or Unigene ID, using another program or website (e.g. source.stanford.edu) in batch mode or one-at-a-time. The list of standard gene names is then submitted in batch to another program or script, or perhaps pasted one-by-one into a website which returns the sequences of upstream regions of the genes. A file containing these collected sequences is in turn scanned against a public database of known transcription factor (TF) binding sites. Yet another program may be used to determine whether the TF patterns returned are over-represented beyond random and thus potentially real regulatory sites for this set of genes. A separate viewing program may be necessary to project these patterns onto the sequences.

Issues for Architecture:

1. Mastering use of several different programs and/or websites may be required, and files must be used to transport data between them.
2. Large amounts of microarray data are manipulated locally.

Use Case 1B –Use of SPLASH in caWorkBench 2.0 to detect upstream conservered sequences:

The researcher loads microarray gene expression data into the caWorkBench 2.0 application using the *Project Management Component* and performs any needed filtering and normalization of the data using appropriate methods from the *Filter and Normalization Components*. The data can be in the form of locally produced files or obtained via direct connections to microarray repositories based on the caArray interface at NCI, Columbia or other sites. He uses the *SOM module* (from the *Analysis Component*) or the *Pattern Discovery Component* to group genes of similar expression pattern together. He then selects a map or a pattern containing an interesting set of co-regulated genes and retrieves their upstream regulatory sequence from the UCSC genome server using the *Promoter Analysis Component*, which relies on the caBIO DAS interface built into caWorkBench 2.0. He then masks repeat and low-complexity regions and runs

SPLASH, from the *Pattern Discovery Component*, to find candidate cis-regulatory motifs among these sequences. The user then uses the *Promoter Analysis Component* again to scan the sequences and motifs for additional known transcription factor binding sites, merging the cis-regulatory motifs discovered by SPLASH with previously known TF binding sites. At each step, intermediate results are stored in the caWorkBench project panel.

Issues for Architecture:

1. caWorkBench 2.0 is loaded from a central server using Java WebStart, thereafter it is kept on the local machine until WebStart detects and downloads a new version.
2. caWorkBench 2.0 possesses a working connection to both NCI caArray- and ArrayExpress-based microarray data repositories, allowing direct browsing of remote datasets. They no longer must be stored locally.
3. caWorkBench 2.0 provides interfaces to numerous other algorithms hosted on servers at Columbia, providing greatly increased processing power for federated users. caGRID interfaces to these are being explored.
4. A history mechanism is present but needs to store more detail.
5. The ability to automate this process requires the development of workflow design and support in caWorkbench.
6. Direct support of MAGE-ML must be developed for caWorkBench 2.0.
7. Native reading of Affymetrix format files is not yet available; data must first be exported to an ASCII format.

Use Case 2a – Current analysis and display of gene expression data on known networks.

The researcher wishes to find regulatory networks among genes by comparing data from microarray gene expression experiments representing different states of a tissue (time points, spatial separation, induced metabolic change etc.). He loads microarray gene expression data into an analysis program. He performs common filtering and normalization steps. He then may perform a measure of differential expression, and/or use a clustering method to find genes with similar expression profiles. A list of genes of interest is produced and written to a file. He then loads the data into a program or website which visualizes expression changes of genes on existing regulatory or metabolic network diagrams, e.g. those available at KEGG. He may also use a program to classify the genes of interest by Gene Ontology (GO) terms and look for overrepresented terms as an indication of a potential regulatory or metabolic pathway.

Issues for Architecture:

1. This method is limited to known pathways and relationships.

2. Some programs can analyze the gene list for potential interactions using a prebuilt database of extracted scientific articles.

Use Case 2b: De novo network prediction (ARACNE) and network visualization by integrating Cytoscape into caWorkBench 2.0

As in Use Case 1b, the researcher loads a large number of microarray gene expression profiles (~100), either from local files or from a caArray-based repository, into the caWorkBench 2.0 *Project Management Component* and performs any needed filtering and normalization of the data using appropriate methods from the *Filter and Normalization Components*. Using the ARACNE algorithm in the *Reverse Engineering Component*, he analyzes the data set and creates an adjacency matrix. He selects a particular gene, e.g. c-Myc, about which to display connected genes, and the maximum number of traversed edges from that gene to reach a displayed node (gene). The network represented in the matrix can then be viewed using the built-in version of Cytoscape or alternatively directly in caWorkBench's own Network Browser. A phenotypic selection is made in the Phenotype Management Component, selecting two classes among the available microarrays. The results of a T-test on each gene in the network graph for the two classes is shown on the graph. Genes that are over-expressed in the first class (cases) are shown in increasingly bright shades of red. Genes that are under-expressed in the first class (cases) are shown in increasingly bright shades of green.
In the *Cytoscape Component*, the researcher clicks on c-Myc and selects its "first neighbors," using the Select Menu. The first neighbors are highlighted and are also broadcast as a selection to the *Gene Management Component*. The list of selected genes can then be used, as in the previous Use Case 1b, to search for common upstream regulatory elements or for any other operation that requires a list of genes. Intersection of the genes with GO families is possible by using the *GO Browser Component*.

Issues for Architecture:

1. Cytoscape (http://www.cytoscape.org, developed by the Institute of Systems Biology and others) was integrated into caWorkbench 2.0 as an external plug-in with minimal effort. Bi-directional communication between the two entities is enabled
2. Computational modules for performing Mutual Information computations are provided with Java Web Start deployed external shared libraries
3. Use of the most accurate network creation options results in heavy computations. An MPI version for use on a parallel cluster architecture is being developed.